Aim : Demonstrate the purpose of feature scalling and show that feature scaling does not effect the distribution of the data

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seborn as sns
```

In [78]:
```
## usecols is supposed to provide a filter before reading the whole DataFrame
df=pd.read_csv("Social_Network_Ads.csv",usecols=["Age","EstimatedSalary","Purchased"])
```

In [79]:
```
df
```

Out[79]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

In [80]:
```
df.head()
```

Out[80]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [81]:
```
df.tail()
```

Out[81]:

	Age	EstimatedSalary	Purchased
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

In [82]:
```
from sklearn.model_selection import train_test_split
```

In [83]:
```
x_train,x_test,y_train,y_test= train_test_split(df.drop("Purchased",axis =1),df["Purchased"],test_size=0.3,random_state=0)
```

In [84]:
```
x_train.shape
```

Out[84]:
```
(280, 2)
```

In [85]:
```
x_test.shape
```

Out[85]:
```
(120, 2)
```

In [86]:
```
from sklearn.preprocessing import StandardScaler
```

In [87]:
```
scaler = StandardScaler()
```

In [88]:
```
scaler.fit(x_train)
```

Out[88]:
```
▾ StandardScaler
StandardScaler()
```

In [89]:
```
x_train_scaled =scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)
```

In [90]:
```
x_train_scaled
```

```
       [ 0.20938504,  1.07558195],
       [ 0.40546467, -0.48604654],
       [-0.28081405, -0.31253226],
       [ 0.99370357, -0.8330751 ],
       [ 0.99370357,  1.8563962 ],
       [ 0.0133054 ,  1.24909623],
       [-0.86905295,  2.26126285],
       [-1.1631724 , -1.5849703 ],
       [ 2.17018137, -0.80415605],
       [-1.35925203, -1.46929411],
       [ 0.40546467,  2.2901819 ],
       [ 0.79762394,  0.75747245],
       [-0.96709276, -0.31253226],
       [ 0.11134522,  0.75747245],
       [-0.96709276,  0.55503912],
       [ 0.30742485,  0.06341534],
       [ 0.69958412, -1.26686079],
       [-0.47689368, -0.0233418 ],
       [-1.7514113 ,  0.3526058 ],
```

In [91]:
```
x_train_scaled = pd.DataFrame(x_train_scaled,columns=x_train.columns)
x_test_scaled = pd.DataFrame(x_test_scaled,columns=x_test.columns)
```

In [92]:
```
scaler.mean_
```

Out[92]:
```
array([3.71666667e+01, 6.95916667e+04])
```

In [93]:
```
x_train
```

Out[93]:

	Age	EstimatedSalary
92	26	15000
223	60	102000
234	38	112000
232	40	107000
377	42	53000
...	...	...
323	48	30000
192	29	43000
117	36	52000
47	27	54000
172	26	118000

280 rows × 2 columns

In [94]:
```
x_train_scaled
```

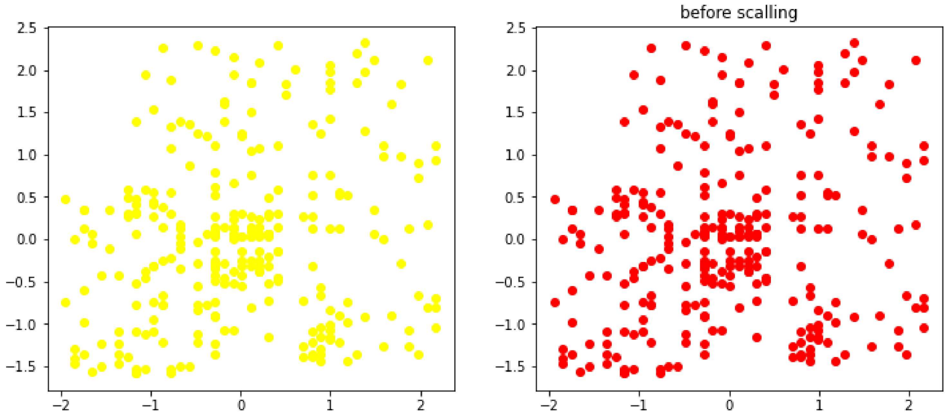
Out[94]:

	Age	EstimatedSalary
0	-1.163172	-1.584970
1	2.170181	0.930987
2	0.013305	1.220177
3	0.209385	1.075582
4	0.405465	-0.486047
...	...	...
275	0.993704	-1.151185
276	-0.869053	-0.775237
277	-0.182774	-0.514966
278	-1.065133	-0.457127
279	-1.163172	1.393691

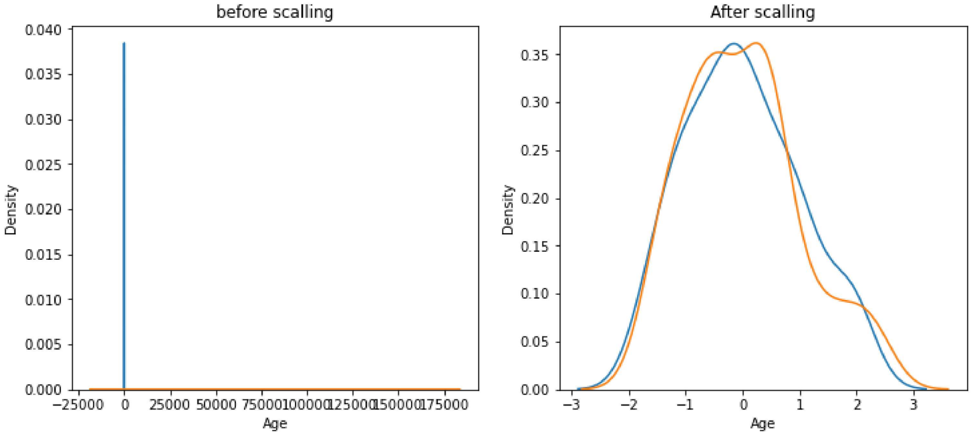
280 rows × 2 columns

In [95]:
```
x_train_scaled = pd.DataFrame(x_train_scaled,columns=x_train.columns)
x_test_scaled = pd.DataFrame(x_test_scaled,columns=x_test.columns)
```

In [96]:
```
from matplotlib import pyplot as plt
fig, (ax1,ax2) = plt.subplots(ncols =2 ,figsize=(12,5))
ax1.scatter(x_train_scaled["Age"],x_train_scaled["EstimatedSalary"],color="yellow")
ax2.scatter(x_train_scaled["Age"],x_train_scaled["EstimatedSalary"],color="red")
ax2.set_title("before scalling")
plt.show()
```



In [98]:
```
from matplotlib import pyplot as plt
fig, (ax1,ax2) = plt.subplots(ncols =2 ,figsize=(12,5))
ax1.set_title("before scalling")
sns.kdeplot(x_train["Age"],ax = ax1)
sns.kdeplot(x_train["EstimatedSalary"],ax=ax1)
ax2.set_title("After scalling")
sns.kdeplot(x_test_scaled["Age"],ax=ax2)
sns.kdeplot(x_test_scaled["EstimatedSalary"],ax=ax2)
plt.show()
```



```
In [ ]:
```